09/143907

Copy

Knobbe Martens Olson & Bear LLP

Intellectual Property Law

550 West C Street
Suite 1200
San Diego CA 92101
Tel 619-235-8550
Fax 619-235-0176
www.kmob.com

Certificate

MAY 24 2006

of Correction

John G. Rickenbrode
jrickenbrode@kmob.com

May 18, 2006

VIA CERTIFIED MAIL

Michael Saunders
President
Legal iGaming, Inc. (A Nevada Corp.)
955 White Drive
Las Vegas, NV 89119

Re: Title: METHOD FOR CONTROL OF GAMING SYSTEMS AND FOR GENERATING RANDOM NUMBERS
U.S. Letters Patent No. 6,986,055
Issued: January 10, 2006
Our Reference: IGAM.3DV1DV1

Dear Michael,

Because Patent Office errors were uncovered during the proofreading of the above-referenced patent, we filed a Certificate of Correction with the U.S. Patent and Trademark Office to officially correct them.

The recorded Certificate of Correction is enclosed. This document should be filed along with the original Letters Patent document forwarded by my letter of February 16, 2006.

MAY 25 2006

If you have questions in connection with the above, please call me

Sincerely,

John Rickenbrode

Enclosure

CERTIFIED MAIL NO.: 7005 0390 0004 3981 1189

2583005:MRF
050806

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 6,986,055
APPLICATION NO. : 09/143,907
ISSUE DATE : August 31, 1998
INVENTOR(S) : Carlson, Rolf E.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the first page, Column 1, Line 1-2, Delete "METHOD FOR GENERATING RANDOM NUMBERS" and insert – METHOD FOR CONTROL OF GAMING SYSTEMS AND FOR GENERATING RANDOM NUMBERS--, therefore.

On the first page, Column 1, under U.S. Patent Documents, Line 1, after "Vasseur" insert -708/250--.

On the first page, Column 2, under U.S. Patent Documents, Line 7, after "Aiello et al." insert –380/46--.

On the first page, Column 2, under U.S. Patent Documents, Line 10, after "Anderson et al." insert –380/46--.

On sheet 3 of 9, box 301, Figure 3, Line 2, Delete "Distribtion" and insert –Distribution--, therefore.

In Column 1, Line 1-2, Delete "METHOD FOR GENERATING RANDOM NUMBERS" and insert --METHOD FOR CONTROL OF GAMING SYSTEMS AND FOR GENERATING RANDOM NUMBERS--, therefore.

In Column 2, Line 20, After "manage risk." Delete "In" and insert –It--, therefore.

In Column 3, Line 13, After "increasing" delete "he" and insert –the--, therefore.

In Column 3, Line 57, Delete "DRAWING" and insert –DRAWINGS--, therefore.

In Column 4, Line 13, Delete "DRAWING" and insert –DRAWINGS--, therefore.

In Column 7, Line 21, Delete "Sytsem" and insert –system--, therefore.

In Column 10, Line 21, Delete "S Ram" and insert –SRAM--, therefore.

2582576
050806

MAILING ADDRESS OF SENDER:

John G. Rickenbrode
KNOBBE, MARTENS, OLSON & BEAR, LLP
2040 Main Street, 14th Floor
Irvine, California 92614

DOCKET NO. IGAM.3DV1DV1

MAY 25 2006

Knobbe Martens Olson & Bear

Issued Patent Proofing Form
Note: **P** = PTO Error **K** = KMOB Error

File#: **IGAM.3DV1DV1**
Proofread By: **Sudhir (04/17/2006)**

US Serial No.: **09/143,907** US Patent No.: **US 6,986,055 B2** Issue Date: **Jan. 10, 2006**
Title: **METHOD FOR GENERATING RANDOM NUMBERS**
PR Instructions: **Whole Patent**

Sr. No.	P/K	Original		Issued Patent		Description of Error
		Page	Line	Column	Line	
1	**P	Page 4 Claims (10/01/2002)	1 – 3 (In the Disclosure) (Preliminary Amendment)	First Page Col. 1 (Title)	1 – 2	Delete "METHOD FOR GENERATING RANDOM NUMBERS" and insert - - METHOD FOR **CONTROL OF GAMING SYSTEMS AND** FOR GENERATING RANDOM NUMBERS - -, therefor.
2	P	Sheet 1 List of references cited by examiner (06/19/2001)	Entry 5 (U.S. Patent Documents)	First Page Col. 1 (U.S. Patent Documents)	1	After "Vasseur" insert - - 708/250 - -.
3	P	Sheet 1 List of references cited by examiner (06/19/2001)	Entry 3 (U.S. Patent Documents)	First Page Col. 2 (U.S. Patent Documents)	7	After "Aiello et al." insert - - 380/46 - -.
4	P	Sheet 1 List of references cited by examiner (06/19/2001)	Entry 2 (U.S. Patent Documents)	First Page Col. 2 (U.S. Patent Documents)	10	After "Anderson et al." insert - - 380/46 - -.
5	K	Sheet 3 Drawings (08/31/1998)	2 Box 301 (Fig. 3)	Sheet 3 of 9 Box 301 (Fig. 3)	2	Delete "Distribtion" and insert - - Distribution - -, therefor.
6	**P	Page 4 Claims (10/01/2002)	1 – 3 (In the Disclosure) (Preliminary Amendment)	1 (Title)	1 – 2	Delete "METHOD FOR GENERATING RANDOM NUMBERS" and insert - - METHOD FOR **CONTROL OF GAMING SYSTEMS AND** FOR GENERATING RANDOM NUMBERS - -, therefor.
7	K	Page 3 Specification (08/31/1998)	23	2	20	After "manage risk." delete "In" and insert - - It - -, therefor.
8	K	Page 5 Specification (08/31/1998)	26	3	13	After "increasing" delete "he" and insert - - the - -, therefor.
9	K	Page 8 Specification (08/31/1998)	1	3	57 (Approx.)	Delete "DRAWING" and insert - - DRAWINGS - -, therefor.
10	K	Page 9 Specification (08/31/1998)	1	4	13 (Approx.)	Delete "DRAWING" and insert - - DRAWINGS - -, therefor.
11	K	Page 16 Specification (08/31/1998)	9	7	21	Delete "sytsem" and insert - - system - -, therefor.
12	K	Page 23 Specification (08/31/1998)	3	10	21 (Approx.)	Delete "S RAM" and insert - - SRAM - -, therefor.

**Note:
In Reference File, **"Request for Corrected Filing Receipt (dated 10/01/2002)"**, **Page 1**, the **Title** is given as:

"METHOD FOR GENERATING RANDOM NUMBERS FOR CONTROL OF GAMING SYSTEMS"

Please Verify.

US006986055B2

(12) **United States Patent**
Carlson

(10) **Patent No.: US 6,986,055 B2**
(45) **Date of Patent: *Jan. 10, 2006**

(54) **METHOD FOR GENERATING RANDOM NUMBERS** 1

(75) Inventor: **Rolf E. Carlson**, Albuquerque, NM (US)

(73) Assignee: **Legal iGaming, Inc.**, Las Vegas, NV (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/143,907**

(22) Filed: **Aug. 31, 1998**

(65) **Prior Publication Data**

US 2003/0028567 A1 Feb. 6, 2003

Related U.S. Application Data

(62) Division of application No. 08/959,575, filed on Oct. 28, 1997, now Pat. No. 6,272,223, which is a division of application No. 08/358,242, filed on Dec. 19, 1994, now Pat. No. 5,707,286.

(51) **Int. Cl.**
H04L 9/32 (2006.01)

(52) **U.S. Cl.** **713/200**; 463/22

(58) **Field of Classification Search** 728/250; 380/46, 44; 463/22, 29; 713/200, 168, 176, 713/189, 193

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,309,509 A * 3/1967 Vasseur 4
4,093,223 A * 6/1978 Wilke et al. 463/31
4,527,798 A * 7/1985 Siekierski et al. 708/250
4,636,951 A 1/1987 Harlick
4,652,998 A * 3/1987 Koza et al. 463/26
5,356,144 A 10/1994 Fitzpatrick et al.
5,398,932 A * 3/1995 Eberhardt et al. 463/29
5,420,928 A * 5/1995 Aiello et al. 2
5,515,307 A * 5/1996 Aiello et al. 380/46
5,707,286 A * 1/1998 Carlson 463/16
5,857,025 A * 1/1999 Anderson et al. 3
5,954,582 A * 9/1999 Zach 708/250
6,099,408 A * 8/2000 Schneier et al. 463/29

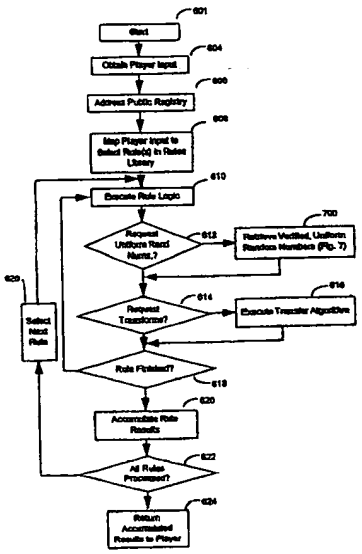
* cited by examiner

Primary Examiner—Ayaz Sheikh
Assistant Examiner—Brandon Hoffman
(74) *Attorney, Agent, or Firm*—Knobbe, Martens, Olson & Bear LLP

(57) **ABSTRACT**

An apparatus for implementing a game having a deterministic component and a non-deterministic component wherein a player uses the game through at least one player interface unit. Each player interface unit generates a player record indicating player-initiated events. A random number generator provides a series of pseudo-random numbers and a rules library stores indexed rules for one or more games. An interface registry stores mapping records where the mapping records are used to associate the player-initiated events to pre-selected rules in the rules library. A control means is coupled to the player interface to receive the output of the player interface unit, coupled to the interface registry, the rules library, and the random number generator. The control means processes the player record and returns an output record to the player interface unit where the output record is determined by executing the game's rules with reference to the pseudo-random numbers and predefined combinatorial algorithms for selecting sets of the pseudo-random numbers.

4 Claims, 9 Drawing Sheets



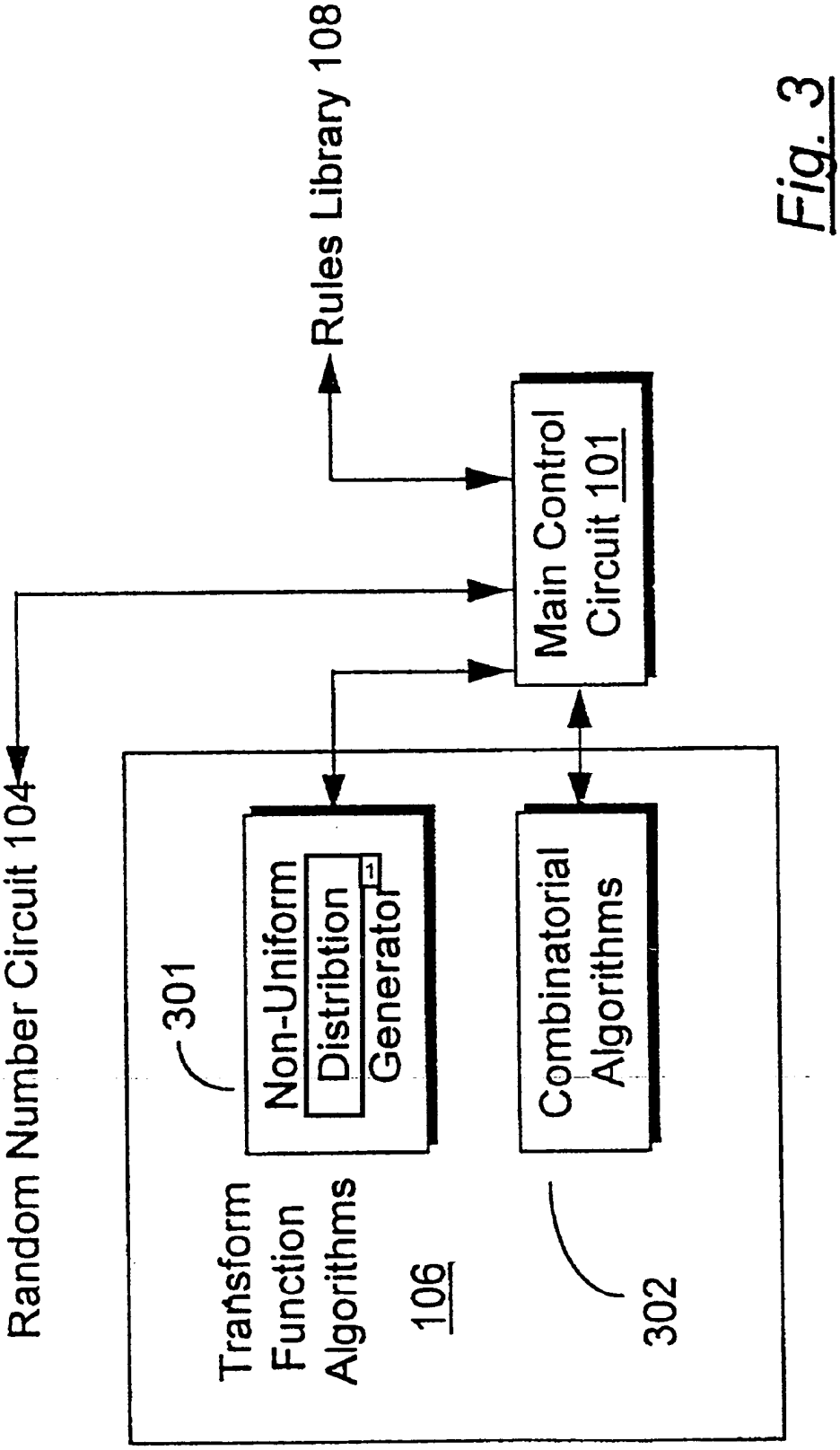

Rules Library 108
Main Control Circuit 101
Random Number Circuit 104
301
Non-Uniform Distribution Generator
Combinatorial Algorithms
Transform Function Algorithms
106
302

Fig. 3

METHOD FOR GENERATING RANDOM NUMBERS

RELATED APPLICATION

This application is a divisional of applicant's U.S. patent application Ser. No. 08/959,575, filed Oct. 28, 1997, now U.S. Pat. No. 6,272,223, entitled "UNIVERSAL GAMING ENGINE," which is a divisional of 08/358,242, the applicant's now U.S. Pat. No. 5,707,286, issued Jan. 13, 1998, entitled "UNIVERSAL GAMING ENGINE" filed Dec. 19, 1994.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates, in general, to gaming machines, and, more particularly, to an electronic gaming engine supporting multiple games and multiple users.

2. Statement of the Problem

Casino gaming has grown rapidly in the United States. Casino gaming is experiencing similar growth throughout the world. An important segment of this developing industry is electronic games. An electronic implementation of a game requires a method for interpreting human actions as they occur within the constraints of the rules as well as the ability to respond with chance events.

Microprocessors allow games that formerly relied on analog devices for generating chance events, such as dice, to be simulated digitally. Simulating a die roll with a computer would seem to be a contradiction because the microprocessor is the embodiment of logic and determinism. With care, however, it is possible to create deterministic algorithms that produce unpredictable, statistically random numbers.

Contemporary games consist of a framework of rules that define the options for how a user or random event generator may change the game state. Play begins with an initial state. Subsequent play consists of user initiated events that trigger the execution of one or more rules. A rule may proceed deterministically or non-deterministically.

Typical games consist of deterministic and non-deterministic rules. A game progresses by the interaction of these rules. There are two sources for non-determinism: player decisions and chance events. In the game of Poker, for example, deciding to replace three instead of two cards in a hand is a player decision that is limited, but not predetermined, by rules. The rules limit the range of options the player has, but within that set of options the player is free to choose. An example of a chance event is the random set of cards received by the poker player. Shuffled cards do not produce a predictable hand.

Other examples that illustrate determinism and non-determinism in gaming are popular casino pastimes such as Blackjack, Keno, and Slot machines. The first Blackjack hand a player receives is two cards from a shuffled deck. The number of cards dealt is two, but the cards could be any from the deck. Keno is essentially a lottery. In Keno, a player attempts to guess twenty balls chosen from a basket of eighty balls. The rules dictate that to participate, a player must fill out a Keno ticket indicating the balls he believes will be chosen in the next round. the selection of balls, however, is a purely random event. Slot machines require the player to pull a handle for each round. Slot wheels stop at random positions.

The non-deterministic problem in most parlor games is random sampling without replacement: given a set of n elements, randomly choose m of them without replacement

where m is less than or equal to n. Although sampling without replacement covers most popular games, it would be easy to conceive of games that required replacement. For example, consider a variant of Keno that replaces each chosen ball before selecting the next ball. Until now, no device is available that services the needs of multiple games by providing algorithms for sampling with and without replacement as well as others such as random permutation generation, sorting, and searching.

A casino player must know the likelihood of winning a jackpot is commensurate with the stated theoretical probabilities of the game. Moreover, the casino would like to payout as little as possible while maximizing the number of their game participants. Because each game sponsored by a casino has a built-in theoretical edge for the house, over time and with repeated play, the house will make money. In other words, the casino does not need to cheat the customer because it has a built-in edge. The customer, who is at a disadvantage in the long run, will want to know the game is fair in order to manage risk. In fact, it is a theoretical fact that bold wagering in Roulette increases a players odds of winning. A player who cannot know the odds of winning cannot formulate a strategy.

Provided that the deterministic rules of a game are implemented correctly, it is essential that the chance events of a game are indeed random. An important subproblem for generating random events is uniform random number generation. If the underlying uniform random number generator does not generate statistically independent and uniform pseudo-random numbers, then either the house or customer will be at a disadvantage. A poorly designed system might favor the house initially and over time turn to favor the player. Certainly the house would not want this situation because it makes revenue projection impossible. Any regulatory body would like to ensure that neither the house nor customer have an advantage beyond the stated theoretical probabilities of the game. In the context of fairly implemented rules, the only way for the house to increase its revenue is to increase the number of players participating in their games.

Typically, an engineer creating an electronic game generates a flow chart representing the rules and uses a random number generator in conjunction with combinatorial algorithms for generating chance events. Representing rules is one problem. Generating chance events to support those rules is another. Creating pseudo-random numbers is a subtle problem that requires mathematical skills distinct from other problems of gaming. In other words, a skilled game programmer may be unable to solve the problems of developing a proper random number generator. Even if given a quality random number generator, problems can occur in hardware implementations that render the generator predictable. One example is using the same seed, or initial state, for the generator at regular intervals and repeatedly generating a limited batch of numbers. Without attending to the theoretical aspects of a uniform random number generator, it is not possible to implement the rules of a game perfectly. The result is a game unfair to the house, players, or both. Hence, there is a need for a gaming system, apparatus, and method that separate the problem of implementing game rules from that of random event generation.

The need for such a device is also evident at the regulatory level. Gaming is a heavily regulated industry. States, tribes, and the federal government have gaming regulatory agencies at various levels to ensure fairness of the games. The gaming regulatory authority certifies that a particular implementations of a game reflects the underlying probabilities.

Because electronic games are implemented in often difficult to understand software, the problem of verifying fairness of a game is challenging. Further, there is little uniformity in the implementation of fundamental components of various games. To determine fairness, the gaming authority subjects each game to a battery of tests. No set of statistical tests performed on a limited portion of the random number generator period can ensure that the generator will continue to perform fairly in the field. The process of testing is both expensive and of limited accuracy. Hence, a regulatory need exists for a uniform, standardized method of implementing games that reduce the need and extent of individual game testing while increasing the liability of detecting and certifying game fairness.

3. Solution to the Problem

The Universal Gaming Engine (UGE) in accordance with the present invention is a gaming apparatus providing a consistent game development platform satisfying the needs of the gaming authority, house, player, and game developer. The UGE separates the problems of developing game rules from the difficulty of producing chance events to support those rules. Functions that are common to a number of games are included in the gaming engine so that they need not be implemented separately for each game. By including basic functions shared by a number of games, hardware costs are greatly reduced as new games can be implemented merely by providing a new set of rules in the rules library and the basic hardware operating the game remains unchanged.

SUMMARY OF THE INVENTION

Briefly stated, the present invention provides a system, apparatus, and method for implementing a game having a deterministic component and a non-deterministic component wherein a player uses the game through at least one player interface unit. Each player interface unit generates a player record indicating player-initiated events. A random number generator provides a series of pseudo-random numbers that are preferably statistically verified by integral verification algorithms and stored in a buffer. Preferably, the random number generator allows seed and key restoration automatically or manually upon power fault.

A rules library stores indexed rules for one or more games. An interface registry stores mapping records where the mapping records are used to associate the player-initiated events to pre-selected rules in the rules library. A control means is coupled to receive the output of the player interface unit, coupled to the interface registry, the rules library, and the random number generator. The control means processes the player record and returns an output record to the player interface unit where the output record is determined by executing the game's rules with reference to the pseudo-random numbers and predefined combinatorial algorithms for selecting sets of the pseudo-random numbers.

BRIEF DESCRIPTION OF THE DRAWING

FIG. **1** illustrates a simplified block diagram of the gaming engine in accordance with the present invention;

FIG. **2** illustrates a block diagram of the pseudo-random number subsystem in accordance with the present invention;

FIG. **3** illustrates the non-uniform distribution generator and combinatorial algorithm subsystems in accordance with the present invention;

FIG. **4** illustrates a main control circuit in accordance with the present invention;

FIG. **5** illustrates in block diagram form implementation of the rules library in accordance with the present invention;

FIG. **6** illustrates a flow chart of a game implementation using the apparatus shown in FIG. **1**;

FIG. **7** illustrates a flow diagram for a second embodiment pseudo-random number distribution system;

FIG. **8** illustrates a multiple player networked implementation in accordance with the present invention; and

FIG. **9** illustrates in graphical form relationships between server speed, queue size, and customer wait times of an apparatus in accordance with the present invention.

DETAILED DESCRIPTION OF THE DRAWING

1. Overview.

FIG. **1** illustrates, in simplified schematic form, a gaming apparatus in accordance with the present invention. The gaming apparatus in accordance with the present invention is also referred to as a "universal gaming engine" as it serves in some embodiments as a platform for implementing any number of games having deterministic and random components. In other embodiments, the universal gaming engine in accordance with the present invention provides a platform that supports multiple players across a network where each player preferably independently selects which game they play and independently controls progression of the game.

Although in the preferred embodiment all of the games discussed are implemented entirely electronically, it is a simple modification to alter the player interface to include mechanical switches, wheels, and the like. Even in mechanically implemented games electronic functions that are performed by the gaming engine in accordance with the present invention are required. Hence, these mechanical machines are greatly simplified using the gaming engine in accordance with the present invention.

Gaming engine **100** is illustrated schematically in FIG. **1**, including major subsystems in the preferred embodiments. Each of the subsystems illustrated in FIG. **1** is described in greater detail below. FIG. **1**, however, is useful in understanding the overall interconnections and functioning of the gaming engine in accordance with the present invention.

Gaming engine **100** performs several basic functions common to many electronically implemented casino games. The most basic of these functions includes interacting with the player to detect player initiated events, and to communicate the state of a game to the player. Gaming engine **100** must process the player initiated event by determining the appropriate rules of the game that must be executed and then executing the appropriate rules. Execution of the rules may require only simple calculation or retrieving information from memory in the case of deterministic rules, or may require access to pseudo-random values or subsets of pseudo-random values in the case of non-deterministic components.

Gaming engine **100** in accordance with the present invention uses a main control circuit **101** to control and perform basic functions. Main control circuit **101** is a hardware or software programmable microprocessor or microcontroller. Alternatively, main control circuit **101** can be implemented as an ASIC device with dedicated logic to perform the required control functions. Main control circuit **101** communicates with player interface unit **102** via interface bus **103**. Player interface unit **102** is a machine having at least some form of display for communicating information to the player and some form of switch (i.e., buttons, levers, keyboard, coin slot, or the like) for communicating information from the player.

games making it cost efficient to use relatively expensive circuitry to provide a high quality random numbered circuit **104**.

Random number generator circuit **201** accepts as input one or more key values which are typically binary values having a fixed relatively large number of bits. For example, the ANSI X9.17 pseudo-random number generator uses 56-bit keys. Random generator circuit **201** also usually accepts a seed value, which is also another large bit binary value. Further, random number generator circuit **201** has a data input or clock input that accepts a continuously variable signal which is conveniently a clock representing date and time. In this manner, each time the signal on the clock or data input changes a new random number is output on line **206**. Random number control circuit stores and provides the key values, seed value, and clock values to random number generator circuit **201**.

A desirable feature in accordance with the present invention is that random number circuit **104** be able to boot up after a power fault (i.e., power is removed from the sytsem 1 using the same seed values, key values, and clock value that existed before the power fault. This feature prevents a player or operator from continually resetting the system or gaining any advantage by removing power from gaming engine **100**. One way of providing this functionality is to buffer the key values, seed values, and clock values in memory within random number control circuit **204** before they are provided to random number generator **201**. After a power on default, circuit **104** can reboot autonomously using the values stored in buffers. Alternatively, new values can be provided via system operator interface **109** to ensure that the output after a power fault is in no way predictable based upon knowledge of output after a prior power fault.

In a preferred embodiment, random number generator circuit operates continuously to provide the series of random numbers on line **206** at the highest speed possible. By continuously, it is meant that random number generator circuit **201** operates at a rate that is not determined by the demand for random numbers by the rest of the system. Random number control circuit **204** provides key values, seed values, and data values to random number generator circuit **201** independently of any processing demands on main control circuit **101** (shown in FIG. 1). This arrangement ensures that random number circuit **104** operates at a high degree of efficiency and is not slowed down by computational demands placed on main control circuit **101**. In other words, the control circuit resources that implement random number control circuit **204** are independent of and usually implemented in a separate circuit from main control circuit **101**.

Random number control circuit **204** accesses one or more verification algorithms **202** via connection **207**. Verification algorithms **202** serve to verify that the raw random numbers on line **206** are statistically random to a predetermined level of certainty. Preferably, verification algorithms **202** include algorithms for testing independence, one-dimensional uniformity, and multi-dimensional uniformity. Algorithms for accomplishing these tests are well known. For example, independence of the pseudo random numbers can be performed by a Runs test. Uniformity can be verified by the Kolmorgorov-Smirnov or K-S test. Alternatively, a Chi-square test verify uniformity. A serial test is an extension of the Chi-square test that can check multi-dimensional uniformity.

Random number control circuit **204** preferably receives and stores a set of raw random numbers from random number generator circuit **201**. The set of raw random numbers can be of any size, for example 1000 numbers. Random number control circuit **204** then implements the verification algorithms either serially or in parallel to test independence and uniformity as described hereinbefore. It may be advantageous to use more than one physical circuit to implement random number control circuit **204** so that the verification algorithms may be executed in parallel on a given set of raw random numbers.

If a set of raw random numbers do not pass one of the verification tests the numbers are discarded or overwritten in memory so that they cannot be used by gaming engine **100**. Only after a batch of numbers passes the battery of verification tests, are they passes via line **208** to verify random number buffer **203**. Buffer **203** is preferably implemented as a first-in, first-out (FIFO) shift register of arbitrary size. For example, buffer **203** may hold several thousand or several million random numbers.

By integrating verification algorithms **202** in a random number circuit **104**, gaming engine **100** in accordance with the present invention ensures that all of the pseudo-random numbers in buffer **203** are in fact statistically random. This overcomes a common problem in pseudo-random number circuits wherein the random numbers are long-term random, but experience short-term runs or trends. These short-term trends make prediction of both the player and casino odds difficult and may create an illusion of unfairness when none in fact exists. The verification algorithms **202** in accordance with the present invention largely eliminate these short-term trending problems and create a pool of random numbers in buffer **203** that are both statistically random and will appear to be random in the short run time period in which both the casino and players operate.

Buffer **203** makes the random numbers available continuously to main control circuit **101**. Main control circuit **101** may access any quantity of the numbers in buffer **203** at a time. Buffer **203** also serves to provide a large quantity of random numbers at a rate higher than the peak generation rate of random number generator circuit **201**. Although it is preferable that random number generator circuit **201** and verification algorithms **202** are processed so as to provide random numbers to buffer **203** at a higher rate than required by gaming engine **100**, short-term bursts of random numbers can be provided by buffer **203** at a higher rate.

3. Transform Function Algorithms.

Transform function algorithms **106** are accessed by main control circuit **101** as illustrated in FIG. **3**. Examples of transform function algorithms **106** are a non-uniform distribution generator **301** and combinatorial algorithms **302**. To execute some rules obtained from rules library **108**, main control circuit **101** may be required to select one or more random values from a non-uniform distribution. Examples of non-uniform distributions are normal distribution, exponential distribution, gamma distribution, as well as geometric and hypergeometric distributions. All of these non-uniform distributions can be generated from the uniform distribution provided by random number circuit **104**.

Rule implementations primarily require that main control circuit **101** access a series of pseudo-random numbers in the context of random set selection and permutations. This subset selection is performed by combinatorial algorithms **302**. The combinatorial algorithms **302** operate on either the uniform number distribution provided directly by random number circuit **104** or the non-uniform distribution provided

by non-uniform distribution generator **301**. In this manner, a game of keno can be implemented by selecting a random 20 from a group of 80.

Another function of the transform algorithms **106** is to scale and center the series of random numbers. For example, a deck of cards includes 52 cards so that the set of random numbers must be scaled to range from 1 to 52. These and similar transform functions are well known.

An advantageous feature of the present invention is that these transform functions can be implemented a single time in a single piece of software or hardware and selectively accessed by any of the games in rules library **108**. This allows a great variety of transform functions to be provided in a cost efficient and computationally efficient manner. The game designer need only provide rules in rules library **108** that access appropriate transform function algorithms **106** and need not be concerned with the details of how the transform function algorithms **106** are implemented. Similarly, a gaming regulatory authority can verify the correctness and fairness of transform algorithms a single time by providing extensive testing. Once the transform functions are verified, they need not be verified again for each game that is implemented in rules library **108**. This independence between the rules programming and the non-deterministic programming result in highly standardized and reliable games while allowing the games designer greater flexibility to design a game in the rules library **108**.

4. Main Control Circuit.

A preferred embodiment of main control circuit **101** is shown in block diagram form in FIG. **4**. Preferably, a micro-controller microprocessor **401** is provided to perform calculations, memory transactions, and data processing. Microprocessor **401** is coupled through bus **103** to player interface unit **102**. Microprocessor **401** is also coupled to player number circuit **104**, transform function algorithms **106**, public interface registry **107**, and rules library **108** through bi-directional communication lines **402**.

In a typical configuration, main control circuit **101** will have a quantity of RAM/SRAM **403**, a quantity of non-volatile memory **404**, and ROM for storing an operating system and boot sequence. ROM **406** operates in a conventional manner and will not be described in greater detail hereinafter. Non-volatile memory **404** is an addressable, preferably random access memory used to store information that is desirably saved even if power is removed from main control circuit **101**. For example, microprocessor **401** may calculate statistics regarding the type of games played, the rate of game play, the rate of number request, or information about the player from player interface unit **102**. The statistics are preferably stored in a non-volatile memory **404** to maintain integrity of the information. Similarly, non-volatile memory **404** may be used to maintain the state of a game in progress on player interface unit **102** so that is power is removed, universal gaming engine **100** can restore player interface unit **102** to the state at which it existed prior to the power outage. This may be important in a casino operation where the casino could incur liability for stopping a game when the player believes a payoff is imminent.

RAM **403** serves as operating memory for temporary storage of rules access from rules library **108** or for storing the operating system for quick access. RAM **403** may also store groups of random numbers while they are being processed by the transform function algorithms as well as address data provided to and accepted from the public interface registry.

It should be understood that main control circuit **101** may be implemented in a variety of fashions using conventional circuitry. While some memory will almost surely be required, the memory may be implemented as RAM, SRAM, EPROM or EEPROM to meet the needs of a particular application. Similarly, the components of main control circuit **101** shown in FIG. **4** may be implemented as a single circuit or single integrated circuit or in multiple circuits or integrated circuits. Additional features may be added to implement additional functions in a conventional manner.

5. Rules Library.

An exemplary embodiment of rules library **108** is illustrated in block diagram form in FIG. **5**. Rules library **108** is preferably implemented as a plurality of volumes of rules where each volume is fixed in a rule EPROM **502–506**. Any number of rule EPROM's can be supplied in rule library **108**. Also, rule EPROM's **502** can be of various sizes. Rule EPROM's **502–506** may be replaced with equivalent memory circuits such as RAM, S RAM 1 or ROM. It is desirable from a gaming regulatory authority standpoint that rule EPROM's **502–506** cannot be altered once programmed so that the rules cannot be changed from the designed rules. This allows the gaming regulatory authority to verify the EPROM rules.

Address logic **501** provides address signals to select one of rule EPROM's **502–506**. Additionally, address logic **501** serves to position a pointer to a specific rule within each rule EPROM **502–506**. As set out herein before, which of rule EPROM's **502–506** is selected as determined by the current game being played as indicated by player interface unit **102** (shown in FIG. **1**). The location of the pointer within a rule EPROM is addressed based upon the current state of the game and the particular user initiated event indicated by player interface unit **102**. The information is conveyed from the user interface unit **102** in a player record that is mapped to rule library **108** by the information in public interface registry **107**.

In practice, a game developer will program a series of rules that dictate the progression of a game in response to user or player initiated events. The rules will also dictate when random numbers are accessed and the type of random numbers which should be accessed (i.e., uniform or non-uniform distributions). Rules will also control payoffs, and place boundaries on the types of player events which will be accepted. The game developer will then burn these rules, once complete, into a rule EPROM, such as rule EPROM's **502–506**. The rule EPROM can then be verified by a gaming regulatory authority, and once approved, be distributed to owners of gaming engines wishing to implement the newly developed game. In order to install the new game, the rule EPROM is installed in rules library **108** and registered in public interface registry **107**. The registration process described hereinbefore provides gaming engine **100** the address information necessary to enable address logic **501** to access a particular rule in rules library **108** and provide that rule on output line **507** to main control circuit **101**.

Although rules library **108** has been described in terms of a plurality of EPROM's **502–506** wherein each EPROM holds one volume of rules pertaining to a particular game, it should be apparent that many other configurations for rules library **108** are possible. Rules can be implemented in a single large memory or in a serial memory such as a tape or disk. Address logic **500** may be integrated in rules library **108**, or may be integrated with main control circuit **101**. Each game may be implemented in a single EPROM or may